Y. PENG
L. CRUICQ
S. BENNANI ZOUBIR
C. ORTIZ BOHORQUEZ
Telecom Bretagne
March 2, 2014

Group 9: c2w Protocol Specification (Version 1.0)

Abstract

This is the first version of c2w protocol specification. The goal of this protocol is to allow c2w application online clients to chat with each other while in the main room, movie rooms or private rooms.

Table of Contents

1. Introduction

The goal of this protocol is to allow c2w application online clients to chat with each other while in the main room, movie rooms or private rooms.

When a new client wants to connect to the server, he has to know the IP Address and the port number of the server and send his unique user−name in the login window.

If the user−name is already used by an other client, this new client will receive an error code indicating that he has to try other names. If login is successful, this client will be directed to the "main room" and request for a movie list and a user list for the GUI display.

For the responses of the movie list and the user list, the server will send back to the client a structured movie list and a structured user list. In the user list, there is one bit for each user who tells the user’s current status, i.e., A(available) or M(Watching movie). In the movie list, every movie is associated with an IP address and a port number that are used to send the video flow. For the sake of simplicity, we assume that the movie list (and asscociated TP address and port number) does not change after the server has started. Every time when there is a new client connects to the server, or when one of the online clients changes his status, the server will send all the "available" users a new user list for updating their GUI display.

The client can send messages to other users in the "main room" by using a text−input box. All the chat messages are displayed in the chat area.

If he presses the "leave" button, he will be disconnected and go back to the login window.

While in the "main room", the user can join one of the movies by clicking on the movie name. The application will then close the current windows and show a new windows with the list of all users in that specific movie room. If the client chat in this movie room, his messages will be sent to all the users in this room. The c2w application uses the Real Time Transfer(RTP) to send the video flow from server to clients. This protocol will NOT deal with sending and receiving videos.

If he presses the "leave" button, he will be disconnected with current movie room and go back to the "main room".

When in the "main room" or in a "movie room" a user clicks on a user-name, he sends a private chat request to the server. If the other user is currently available of private chat, i.e., he doesn't have a private chat room, both side of these two users opens a private room window. We note that a user can only have one private chat at once. He will get error message if he tries to open a second private chat window. He will receive error code from the server if the user he demanded to chat is already engaged in private chat.

If the user presses the "leave" button of the private room, the corresponding window is closed and the private chat is terminated.

For the sake of simplicity, the packet format is the same for all the messages.

This protocol is able to be implemented either with TCP or UDP.

2. Packet format

```
 0               1               2               3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Sequence    |     Type      |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                                                               .
.                             Data                              .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1

When this protocol is implemented by using UDP, the Data field should not exceed 65523 bytes. This is because the length of UDP datagram is limited in 65535 bytes (including an 8 bytes header). Since the header of the c2w protocol is 4 bytes, we can easily calculate the maximum length of the Data field.

When it is implemented by using TCP, the Data field is limited by the Length field, so the maximum should be 65535 bytes.

2.1. Sequence field

This field contains an integer between 0 and 255. It is initialed with the value 0. It is then increased 1 for each time a message is sent. When it reaches 255, it goes back to 0 for the next message and continues increasing for the following messages.

This field is especially designed for the acknowledgment of receiving messages. That is, the value of this field should be the sequence number of the correct packet. For example, when the sequence of a message received is not the one expected, the machine should wait until the correct sequence arrives. In the other side who send this message, if he doesn’t receive the acknowledgment sequence which indicates that the message is well delivered since 2 seconds, he should resend this message so that the conversation between them can continue.

When you implemented this protocol by using UDP, you should particularly pay attention to this field because the datagrams arrived are unordered.

2.2. Length field

This is the length of the data, its value cannot exceed 65535.

2.3. Type field

2.3.1. Message types from client to server

Type	Message Name	Data
0000 0001	loginRequest	user_name
0000 0010	movieListRequest	-
0000 0011	mainRoomUserListRequest	-
0000 0100	movieRoomUserListRequest	movie_name
0000 0101	sendMsgInMainRoom	msg
0000 0110	sendMsgInMovieRoom	msg
0000 0111	sendMsgInPrivateRoom	msg
0000 1000	privateChatRequest	user_name
0000 1001	movieRequest	movie_name
0000 1010	leaveMainRoomRequest	-
0000 1011	leaveMovieRoomRequest	-
0000 1100	leavePrivateRoomRequest	-
1000 0000	userListReceived	-
1000 0001	msgReceived	-
1000 0010	privateChatRequest	-

Table 1: Message from client to server

loginRequest: When a client login, he has to create a user name and send it to the server. This client will then receive either a positive response LOGIN_SUCCESS or a negative response USERNAME_EXIST from the server. If login succeeds, the client can send movieListRequest and mainRoomUserListRequest for its GUI display.

movieListRequest: This message is sent every time after a user is logged in. For the response, the server will send a structured list of movies. In this request, the data field doesn’t contain any message.

mainRoomUserListRequest: When a client is successfully logged in or when he leaves a movie room and back to main room, he sends this request packet for the GUI display. In this request, the data field doesn’t contain any message.

movieRoomUserListRequest: When a client entres in a movie room, he sends this request for the GUI display. The server will send the user list of the movie room that the client demands. In this request, the data contains the name of the movie room.

sendMsgInMainRoom: This message can only be sent by the clients in the main room. The message will be forwarded by the server to all the clients in the main room. The data field will contain the chat message, and the Length field will indicate the length of the data field.

sendMsgInMovieRoom: This packet can only be sent by the clients in a movie room. The message then will be forwarded by the server to all the clients in the same movie room. The data field contains the chat message, and the Length field indicates the length of the data field.

sendMsgInPrivateRoom: When a private room is opened, the client can send private messages to the other side of this private room with this type of message. This message is sent to the server and then forwarded to the other client. The data field contains the chat message, and the Length field indicates the length of the data field.

privateChatRequest: When a client clique a user on the user list in the main room or movie room, a privateChatRequest will be sent to the server for applying a private room between this client and the user he wants to chat. In this request, the data field contains the user name of the user requested to chat.

movieRequest: The message is a request with the name of a movie to the server to join this movie room. The server will response with the user list in the selected movie room after modifying the client’s status in the database. In the same time, the server will inform the users in this movie room that a new user is in by sending a new user list. After that, the server will send to all the available clients in the main room a refreshed user list.

leaveMainRoomRequest: Sends a request to the server, asking it to end the connection. The server will then acknowledge with disconnectSuccess. The server will send the sendMainRoomUserList message to all the clients in the main room, so that the other users can update the GUI display.

leaveMovieRoomRequest: Sends a request to the server, asking it to leave the movie room. The server will then acknowledge with removedFromMovieRoom. The server will then send the new user list of the movie room to the rest of the users in the movie room, and send the full user list to all the availble users in the main room.

leavePrivateRoomRequest: This message is sent when a user clics the leave button of a private room. The server will inform the other user in the private room to quit and send to the client a privateRoomReleased as an ACK message.

privateChatStar: Acknoledgement message for the accomplishment of establishing a private chat.

userListReceived: Acknoledgement message for receiving a user list.

msgReceived: Acknoledgement message for receiving a chat message. The chat message can be from the main room, a movie room, or a private room.

2.3.2. Message types from server to client

Type	Message Name	Data
0000 0000	loginResponse	status_code
0000 0001	movieListResponse	movie_list
0000 0010	sendMainRoomUserList	user_list
0000 0011	sendMsgInMainRoom	msg
0000 0100	sendMsgInMovieRoom	msg
0000 0101	sendMsgInPrivateRoom	msg
0000 0110	privateChatResponse	status_code
0000 0111	sendMovieRoomUserList	user_list
1000 0001	movieResponse	status_code
1000 0010	removedFromMovieRoom	-
1000 0011	disconnectSuccess	-

Table 2: Messages from server to client

loginResponse: When a loginRequest is sent to the server, the latter would respond with either LOGIN_SUCCESS or USERNAME_EXIST as a status-code in the data field.

movieListResponse: This response delivers a structured movie list to the client. The data field contains the movie list.

sendMainRoomUserList: When a client sends a mainRoomUserListRequest, he receives a list containing the names of the users in the main room (user_list). This list is contained in the data field.

sendMsgInMainRoom: This packet is transferred from the server to the available clients in the main room. It contains the message originally sent by the client and the length field indicates the length of the data.

sendMsgInMovieRoom: This packet is transferred from the server to the clients in the movie room from which the message is sent. It contains the message originally sent by the client and the length field indicates the length of the data.

sendMsgInPrivateRoom: When a client send a message in a private room, the server forwards this message to the other side by using this packet. It contains the message originally sent by the client and the length field indicates the length of the data.

privateChatResponse: When the server receives a privateChatRequest, it has to check if the user demanded is available. If yes, the server responds a PRIVATE_ROOM_AVAILABLE in the data field, otherwise a PRIVATE_ROOM_UNAVAILABLE.

sendMovieRoomUserList: When a client enters into a movie room, the server send him the current user list of the movie room. Also, each time when the user list of a movie room is changed, the reffered users will receive this message for the requiement of refreshing the GUI display.

movieResponse: This is an acknowledgment message from the server to indicate if the user has successfully joined in the movie room. If the movie room is not full, the status_code in this message will be MOVIE_JOIN_SUCCESS, otherwise USER_FULL. The capacity of a movie room depands on different server machines.

removedFromMovieRoom: This is an acknowledgment message from the server which indicates that the user has successfully quit the movie room.

disconnectSuccess: This is an acknowledgment message from the server which indicates that the user has successfully disconnected.

2.4. Data field

2.4.1. Empty data field

When the data field is empty, it means the packet only contains a header where length is zero.

2.4.2. Status code

The status_code (8 bits) returned from server to client

Message Type	status name	status_code
loginResponse	LOGIN_SUCCESS	0000 0000
	USERNAME_EXIST	0000 0001
privateChatResponse	PRIVATE_ROOM_UNAVAILABLE	0001 0000
	PRIVATE_ROOM_AVAILABLE	0001 0001
movieResponse	MOVIE_JOIN_SUCCESS	0010 0000
	USER_FULL	0010 0001

Table 3: Status code returned from server

2.4.3. User list

A structured user_list returned from the server to the client is a list of user_names.

```
 0               1
 0 1 2 3 4 5 6 7 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|   Length    |                    user_name                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|   Length    |                    user_name                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.               .                                                .
.               .                                                .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2

|S| (1 bit) refers to the user status: 1 if available, 0 if in movie room. Length(7 bits) indicates the length of the user_name, which varies from 1 to 127 bytes.

When this list is sent by the sendMovieRoomUserList, the |S| is always 1.

2.4.4. Movie list

A structured movie list returned from server to client is as below:

```
0     1     2     3     4     5     6     7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     IP Address        |    Port    |Length| Movie Name |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     IP Address        |    Port    |Length| Movie Name |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                       .            .      .            .
.                       .            .      .            .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3

Each movie is associated with an IP address and a port number. The length indicates the length of the movie name, which varies from 1 to 255 bytes.

3. Reliability

When the c2w protocol is realized by using UDP, packets sent by the clients and the server may be lost. In this case, resending the lost packets has to be concerned. For the reason that the server might be overloaded by these resent packets, clients have to wait at least 2 seconds before resending a request.

4. Server configuration

The server must contain a list of all the connected clients with their user names and status. If a client is in a movie room, this list should contain the name of this movie room. If a client is engaged in a private chat, this list should include the name of the user with whom he is chatting.

5. Example scenario

For the following scenarios, the upper side of an arrow are the type of the message, and the lower side of an arrow is the data contained in the Data field. If there is nothing under an arrow, it means that the Data field of this message is empty.

5.1. connect to server

```
                     Client                               Server
                       |                                     |
                       |           loginRequest              |
                       |------------------------------------>|
                       |          (exist username)           |
                       |                                     |
                       |           loginResponse             |
                       |<------------------------------------|
                       |           (USERNAME_EXIST)          |
                       |                                     |
                       |           loginRequest              |
                       |------------------------------------>|
                       |          (unique username)          |
                       |                                     |
                       |           loginResponse             |
                       |<------------------------------------|
                       |          (LOGIN_SUCCESS)            |
                       |                                     |
                       |           movieListRequest          |
                       |------------------------------------>|
                       |                                     |
                       |                                     |
                       |          movieListResponse          |
                       |<------------------------------------|
                       |            (movie_list)             |
                       |                                     |
                       |        mainRoomUserListRequest      |
                       |------------------------------------>|
                       |                                     |
                       |                                     |
                       |         sendMainRoomUserList        |
                       |<------------------------------------|
                       |            (user_list)              |
                       |                                     |
                       |          userListReceived           |
                       |------------------------------------>|
                       |                                     |
```

Figure 4

In this scenario, the client is logging into the server. First he tries a user name which is already used by another user and the server responds with a negative status_code(USERNAME_EXIST). So the client tries again with an other user name which is unique and then he receives a positive answer(LOGIN_SUCCESS). After opening the main room window, the client needs to display the movie list and all the users on the server with their status. He sends two requests to the server, who responds with a list of movies and a list of all the users. The mainRoomUserListRequest is also considered as an ACK message of well receiving the movie_list.

To ensure that the user_list is well delivered, the client must send an userListReceived as an ACK packet.

5.2. Chat in main room

```
         Client                      Server                Other Clients
           |    sendMsgInMainRoom      |                         |
           |-------------------------->|                         |
           |     (chat message)        |                         |
           |                           |                         |
           |     sendMsgInMainRoom     |                         |
           |<--------------------------|  sendMsgInMainRoom      |
           |      (message)            |------------------------>|
           |                           |     (message)           |
           |      msgReceived          |                         |
           |-------------------------->|                         |
           |                           |    msgReceived          |
           |                           |<------------------------|
```

Figure 5

When the client sends a chat message in the main room, the message will be firstly sent to the server and then be forwarded to all the available users in the main room, including the client himself, who treat this echo from the server as an ACK packet of receiving the previous message sent by him. To ensure that the all the referred clients have received this message, they have to respond to the server with msgReceived as an ACK packet. If the server doesn’t receive this ACK packet of one of the users after 2 seconds, it will resend the chat message.

If the client sends this message in a movie room, the message will be delivered in the same way except that the users who receive this message are replaced to the users in the concerned movie room.

5.3. Join a movie room

```
                     Client                                Server
                       |                                      |
                       |            movieRequest              |
                       |------------------------------------->|
                       |            (movie_name)              |
                       |                                      |
                       |          movieResponse               |
                       |<-------------------------------------|
                       |       (MOVIE_JOIN_SUCCESS)           |
                       |                                      |
                       |     movieRoomUserListRequest         |
                       |------------------------------------->|
                       |           (movie_name)               |
                       |                                      |
                       |       sendMovieRoomUserList          |
                       |<-------------------------------------|
                       |            (user_list)               |
                       |                                      |
                       |          userListReceived            |
                       |------------------------------------->|
                       |                                      |
```

Figure 6

The client select a movie from the movie list. This action triggers a movieRequest, and the client wait for a positive response from the server so that he can continue with a movieRoomUserListRequest for the movie room window display.

To ensure that the user_list is well delivered, the client must send an userListReceived as an ACK packet.

5.4. Start a private chat

```
           Client A                     Server                 Client B
               |    privateChatRequest     |                         |
               |--------------------------->|                         |
               |(unavailable user_name)    |                         |
               |                           |                         |
               |     privateChatResponse   |                         |
               |<---------------------------|                         |
               | (PRIVATE_ROOM_UNAVAILABLE) |                         |
               |                           |                         |
               |     privateChatRequest    |                         |
               |--------------------------->|                         |
               |   (available user_name)   |   privateChatRequest    |
               |                           |------------------------>|
               |                           |     (user_name)         |
               |    privateChatResponse    |                         |
               |<---------------------------|  privateChatStarted     |
               |  (PRIVATE_ROOM_AVAILABLE)  |<------------------------|
               |                           |                         |
               |   privateChatStarted      |                         |
               |--------------------------->|                         |
               |                           |                         |
```

Figure 7

The client A select a user from the user list of the main room (or a movie room). If the user selected to chat is currently available, which means nobody is now chatting with him, the server returns an positive response to indicate that he can open a private chat room to start the conversation. In the same time, the user requested to chat (Client B) will receive a privateChatRequest with the user name of the Client A.

Both side need to send an ACK message(privateChatStarted) to the server to ensure that the private chat is established.

5.5. Chat in a private room

```
           Client A                     Server                  Client B
               |   sendMsgInPrivateRoom    |                         |
               |-------------------------->|                         |
               |       (message)           | sendMsgInPrivateRoom    |
               |                           |------------------------>|
               |                           |        (message)        |
               |                           |                         |
               |                           |     msgReceived         |
               |                           |<------------------------|
               |       msgReceived         |                         |
               |<--------------------------|                         |
               |                           |                         |
```

Figure 8

When client A sends a message to client B, this message is forwarded by the server to Client B. Client B has to answer with an ACK packet (msgReceived), then the server inform client A that the message is received.

5.6. Leave private room

```
            Client A                    Server                  Client B
                | leavePrivateRoomRequest |                         |
                |------------------------>|                         |
                |                         | leavePrivateRoomRequest |
                |                         |------------------------>|
                |                         |                         |
                |                         | leavePrivateRoomRequest |
                |                         |<------------------------|
                | privateRoomReleased     |                         |
                |<------------------------|  privateRoomReleased    |
                |                         |------------------------>|
                |                         |                         |
```

Figure 9

When client A clicks the leave button of the private room, he sends a request to inform the server that he closed the private chat window. The server need to tell client B that client A quits, so when client B receives this message, he sends the same request to the server in order to properly close the conversation. To ensure the leavePrivateRoomRequest is well received, the server must send a privateRoomReleased as an ACK packet after releasing the conversation conserved in the server.

5.7. Disconnect from server

```
                    Client A                                  Server
                      |                                        |
                      |         leaveMovieRoomRequest          |
                      |--------------------------------------->|
                      |                                        |
                      |        removedFromMovieRoom            |
                      |<---------------------------------------|
                      |                                        |
                      |          leaveMainRoomRequest          |
                      |--------------------------------------->|
                      |                                        |
                      |          disconnectSuccess             |
                      |<---------------------------------------|
                      |                                        |
```

Figure 10

If the client wants to leave the movie room, he sends a request and closes the window of the movie room and opens the window of the main room. On the server side, it removes this user from the movie room's user list and changes the status of the user from M(movie) to A(availble). The server sends an ACK to the client after these operations.

When the client quits the main room, he sends a leaveMainRoomRequest The server will remove the this user from the user list and send an ACK packet to inform the user that he is disconnected. Then the client closes the main room window and opens the login window.

6. Conclusion

This the end of the c2w protocol specification.

Authors' Addresses

Yuancheng PENG
Telecom Bretagne
Brest,
France

Email: yuancheng.peng@telecom-bretagne.eu

Lucie CRUCQ
Telecom Bretagne
Brest,
France

Email: lucie.crucq@telecom-bretagne.eu

Saad BENNANI ZOUBIR
Telecom Bretagne
Brest,
France

Email: saad.bennanizoubir@telecom-bretagne.eu

Cuauhtemoc ORTIZ BOHORQUEZ
Telecom Bretagne
Brest,
France

Email: cuauhtemoc.ortizbohorquez@telecom-bretagne.eu